8-31-2018

Decoupling Information and Connectivity via Information-Centric Transport

Hila Ben Abraham
Washington University in St Louis, hila@wustl.edu

Jyoti Parwatikar
Washington University in St. Louis, jp@wustl.edu

John DeHart
Washington University in St. Louis, jdd@wustl.edu

Adam Drescher
Washington University in St. Louis, adrescher@wustl.edu

Patrick Crowley
Cisco Systems and Washington University in St. Louis, pcrowley@wustl.edu

Decoupling Information and Connectivity via Information-Centric Transport

Hila Ben Abraham
Washington University in St. Louis
hila@wustl.edu

Jyoti Parwatikar
Washington University in St. Louis
jp@wustl.edu

John DeHart
Washington University in St. Louis
jdd@wustl.edu

Adam Drescher
Washington University in St. Louis
adrescher@wustl.edu

Patrick Crowley
Cisco Systems &
Washington University in St. Louis
pcrowley@wustl.edu

ABSTRACT

The power of Information-Centric Networking (ICN) architectures lies in their abstraction for communication — the request for named data. This abstraction promises that applications can choose to operate only in the information plane, agnostic to the mechanisms implemented in the connectivity plane. However, despite this powerful promise, the information and connectivity planes are presently coupled in today's incarnations of leading ICNs by a core architectural component, the forwarding strategy. Presently, this component is not sustainable: it implements both the information and connectivity mechanisms without specifying who should choose a forwarding strategy — an application developer or the network operator. In practice, application developers can specify a strategy only if they understand connectivity details, while network operators can assign strategies only if they understand application expectations.

In this paper, we define the role of forwarding strategies, and we introduce Information-Centric Transport (ICT) as an abstraction for cleanly decoupling the information plane from the connectivity plane. We discuss how ICTs allow applications to operate in the information plane, concerned only with namespaces and trust identities, leaving network node operators free to deploy whatever strategy mechanisms make sense for the connectivity that they manage. To illustrate the ICT concept, we demonstrate ICT-Sync and ICT-Notify. We show how these ICTs 1) enable applications to operate regardless of connectivity details, 2) are designed to satisfy a predefined set of application requirements and are free from application-specifics, and 3) can be deployed by network operators where needed, without requiring any change to the application logic.

CCS CONCEPTS

• **Networks → Network design principles**; *Layering*; *In-network processing*;

ICN '18, September 21–23, 2018, Boston, MA, USA

ACM ISBN 978-1-4503-5959-7/18/09.
https://doi.org/10.1145/3267955.3267963

KEYWORDS

ICN, NDN, ICT, Forwarding Strategy

ACM Reference Format:
Hila Ben Abraham, Jyoti Parwatikar, John DeHart, Adam Drescher, and Patrick Crowley. 2018. Decoupling Information and Connectivity via Information-Centric Transport. In *ICN '18: 5th ACM Conference on Information-Centric Networking (ICN '18), September 21–23, 2018, Boston, MA, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3267955.3267963

1 INTRODUCTION

Advocates for Information-Centric Networking (ICN) architectures argue that the IP underlying telephony-inspired abstraction, in which pairs of addressed endpoints must establish a connection to communicate (i.e, a telephone call), does not comply with the requirements of today's Internet. Hence, to fully address IP's challenges, ICN must use another abstraction — the request for named data [1].

Although the request for named data abstraction was introduced and popularized by the World Wide Web and HTTP, HTTP-based applications are bound to the channel abstraction and must respond to connectivity events, such as a change of an IP address or connectivity loss. Therefore, the request for named data abstraction in the network layer not only provides native support for data dissemination and content-centric security models, but also introduces a new and powerful promise to ICN applications: the freedom to stay in the *Information Plane*, free from connectivity details. In other words, the application can be concerned only with data namespaces and trust identities of data producers and consumers, without worrying about the network characteristics.

However, despite this powerful promise, recent work [2–4] has shown that the two leading ICN architectures, Named Data Networking (NDN) [5] and Content Centric Networking (CCN) [6], presently couple applications with the details of connectivity. Specifically, a central architectural component in NDN and CCN, the forwarding strategy, binds applications to the details of connectivity in an unsustainable way. To see why, consider that a forwarding strategy dynamically determines the answers to questions such as: 1) If routing rules permit multiple, equivalent next-hops, which should be chosen?, 2) If a link goes down or if a packet times out, should a packet be retransmitted on some other next hop?, 3) Should a packet be "broadcast" to all eligible next hops?

Clearly, the answers to such questions rely on connectivity characteristics, such as where in the network the node is located (e.g,

core or edge), the number and type (e.g., wired or wireless) of next hop links available, and the dynamics of the network (e.g., static or mobile). However, such strategy questions are also meaningful for application developers, because they are intrinsic to information flow and hence to application structure. For example, if an application is not certain if and when a retransmission will take place, then the application must be structured to retransmit according to the specifics of the forwarding strategy [4].

Moreover, it is unclear who selects the forwarding strategy. Presently, the two software implementations of NDN and CCN, the NDN Forwarding Daemon (NFD) [7] and Community ICN (CICN) [6], allow application developers to specify a forwarding strategy and associate its strategy choices with their namespaces. However, while a developer can pair a forwarding strategy with its application namespace in the localhost, forwarding strategies are assigned within nodes interior to the network by the operators of those specific nodes. Therefore, neither application developers nor network operators can optimally select a forwarding strategy, because the right choice depends on knowledge that neither party alone possesses in its entirety. This difficulty can be mitigated in isolated environments where application developers also operate the entire network, such as with the global NDN testbed [8]. However, in general, the forwarding strategy component unsustainably couples information and connectivity.

To clearly define the scope of the paper, we argue that the problem lies in ICN having one architectural component that both reconciles application and network considerations and manages the interests of both applications and network operators. Therefore, the goal of this paper is to propose architectural modifications to ICN that allow effective decoupling of information and connectivity mechanisms. We do this by specifying, for the first time, the role of forwarding strategies in the ICN architecture, and by proposing a new abstraction for information-oriented mechanisms, named Information-Centric Transport (ICT).

We define an ICT to be an abstraction and a communications mechanism designed to support a specific, but broadly applicable, set of application requirements. An ICT consists of an end-point API and an optional intermediate service. While forwarding strategies implement connectivity-oriented mechanisms, ICTs implement information-oriented mechanisms and deal with namespaces and trust identities. We show how the placement of an intermediate ICT service in the network can provide sustainable communications to different applications, without deploying or relying on any application-specific code in the network, and while keeping the intermediate ICT and strategy mechanisms transparent to the application.

Once an application developer relies on an ICT API for information dissemination, then forwarding strategy choices can be made unilaterally by network node operators, provided that those choices faithfully implement the ICT. Therefore, ICT decouples the information and connectivity planes in a general-purpose way, and solves the problem of how to sustainably make forwarding strategy choices.

To illustrate our contribution concretely, we implemented two ICTs: ICT-Sync, an ICT for sync-based applications, and ICT-Notify, an ICT for notification-based applications. We show how these ICTs implement different application requirements and provide communications for their applications under a range of connectivity scenarios, where IP tools and native NDN applications fail or do not work well. For instance, links may be sufficiently intermittent that there is never a synchronous end-to-end path available between a consumer and a producer.

It is important to note that the paper's principal goal is to discuss the concept and placement of the ICT abstraction in the architecture, and not the specifics of different ICT mechanisms. We use ICT-Sync and ICT-Notify to demonstrate what these mechanisms might look like, but the details of specific ICT mechanisms are not the main focus of this work. Moreover, while ICT-Sync and ICT-Notify are designed to satisfy different application requirements, they both implement best-effort mechanisms and are not designed to address specific performance properties. In the real world, mechanisms for performance properties intrinsically couple information-level and connectivity-level considerations. Therefore, understanding if and how such mechanisms can be implemented while maintaining architectural clarity within the content-centric approach is another interesting open question.

The contributions of this paper are as follows:

- We specify, for the first time, the architectural role of forwarding strategies in ICN.
- We introduce Information-Centric Transport and discuss its placement in the ICN architecture.
- We demonstrate the ICT abstraction by discussing two ICTs: ICT-Sync, an enhanced version of NDN sync, as an ICT for sync-based applications, and ICT-Notify, a new NDN notification mechanism for push-based NDN applications.
- We show how ICT-Sync and ICT-Notify allow applications to operate in unstable environments with lossy and intermittent links, even when there is never a synchronous end-to-end path between a producer and a consumer.
- We discuss the properties of an intermediate ICT service as a mechanism that should remain transparent to applications, and that should not include any application-specifics.

2 BACKGROUND AND RELATED WORK

Information-Centric Networking (ICN) is a future internet architecture that follows the content-centric approach. While the traditional IP architecture uses addresses to identify the source and destination of every exchanged packet, the content-centric approach uses names in its Interest and Data packets to request and retrieve content items. Named Data Networking (NDN) [5] and Content Centric Networking (CCN) [6] are two ICN architectures, and each has its own implementation of an ICN forwarder [6, 7].

In ICN, consumers express an Interest packet to request a content by its name, and producers respond with Data packets. ICN uses three data structures to forward and retrieve Interests and Data: a Forwarding Information Base (FIB) table, which consists of prefixes and potential faces (upstreams) that can satisfy name requests; a Content Store (CS), which keeps a replica of a Data packet forwarded back to the consumer; and a Pending Interest Table (PIT), which records and aggregates faces of incoming Interests to be used when a Data packet is sent back to the consumer.

When a router receives an Interest packet, it first looks for the requested Data in its CS. If there is no match, the router searches

its PIT for a record of an unsatisfied Interest with the same name. If no PIT entry matches the incoming Interest's name, the router searches the FIB for a list of potential upstreams that can satisfy the requested Interest. If the list of upstreams contains more than one potential face, the *forwarding strategy* component decides to whom the Interest should be forwarded, and the router records the forwarded Interest in its PIT. When an Interest can be satisfied by a content found in a router's CS or generated by a producer, a Data packet is sent back to the consumer(s) by following the breadcrumbs recorded in the PIT.

2.1 Forwarding Strategies

In past years, the forwarding strategy module has been demonstrated to be a key architectural component of the ICN architecture. Unlike IP, the FIB in ICN can contain more than one possible next hop for a namespace. In such cases, the forwarding strategy component is required not only to decide to whom to forward the Interest, but also to determine whether and when to send or retransmit an unsatisfied Interest [2, 5, 9].

ICN allows different forwarding strategies to co-exist, and therefore, supports a range of different forwarding algorithms for different connectivities. Moreover, both NDN and CCN support the name-based strategy selection design, in which a namespace can be paired with a specific forwarding strategy, and therefore an application can control its forwarding behavior. This allows ICN to support flexible stream-based forwarding and to provide different in-network mechanisms for different applications.

The frequently cited "Named Data Networking" paper [5] describes this core architectural component as "the key to NDN's resiliency and efficiency". However, despite the central role the forwarding strategy plays, its architectural role has not been well understood, and it remains an underspecified piece in the ICN architecture. Figure 1 shows the building blocks of NDN, with the strategy layer residing between the MAC layer and the Named Data layer [5].

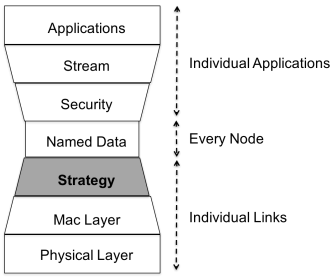

Figure 1: NDN Building Blocks as Described in [5]

3 ON THE ROLE OF FORWARDING STRATEGIES

By studying the role of forwarding strategies in current implementations of CICN [6] and NFD [10], and by reading related works, one arrives at to two contradictory assumptions. 1) A forwarding strategy can be paired with an application namespace, and 2) A forwarding strategy can address desired connectivity characteristics. If both assumptions are true, then a forwarding strategy couples both applications and network mechanisms, and therefore introduces challenges for an application developer who 1) cannot guarantee that the same strategy is used everywhere along the path(s) to the producer, 2) must modify its application whenever the strategy behavior is updated in future versions, and 3) should potentially develop different versions of its application to support its operation in different network environments. In other words, coupling the mechanisms of both network and applications in the forwarding strategy module does not scale and makes it hard to develop ICN applications.

3.1 The Information and Connectivity Planes

To clearly define the forwarding strategy's role, we look at the ICN architecture as a whole and identify the abstraction it aims to provide and how it provides it.

We argue that ICN applications operate in the *Information Plane*, and the network operates in the underlying *Connectivity Plane*. To see why, consider that ICN applications ask for data by name, and the network must find and retrieve that data. But how does the network do that? Unlike IP, ICN is channel-less and consist of different hop-by-hop mechanisms to find requested data. In practice, there is always an actual, real-world connectivity present — e.g., the collection of one or more connectivity options, including WiFi links, Ethernet links, TCP channels, BT, and UDP multicast. Because the properties of these different connectivities differ so widely, the best choice of mechanism in any given circumstance may depend strongly on the specific connectivity available.

In the TCP/IP model, an HTTP name is translated to an IP address at the endpoint. However, in the content-centric approach, the name used by an ICN application is also used by the ICN network as the identifier of core network operations, including name-based Interest forwarding, name-based routing, and name-based caching. The great benefit of using the same identifier in both the information and connectivity planes is that the application can operate in an agnostic way, without having to worry about the specifics of connectivity. By contrast, HTTP-based applications can break when 1) devices change IP addresses, 2) devices have and try to use multiple concurrent interfaces, and 3) Internet connectivity is lost. It is true that HTTP-based applications can implement mechanisms to respond to such events, but they are still coupled with the events' occurrences.

We use a set of illustrative questions to discuss how ICN uses the request for named data abstraction in the network layer. Suppose that a consumer application asks: "What is the content for this name?" Here, the consumer does not specify where the content can be found, or how to get it. In theory, the consumer's question can be answered simply by broadcasting it until someone replies with the requested named data. However, broadcasting is an expensive network operation, and flooding the network is not a scalable solution. Therefore, this simple abstraction must somehow be translated by the network to a practical mechanism that can efficiently find and retrieve the requested content. In other words, ICN must somehow move from the information plane to the connectivity plane.

We illustrate the process of moving from the information plane to the connectivity plane by asking two more questions: 1) "Who

might have the content for this name?" and 2) "What is the most efficient way to retrieve it"? These two questions should be answered differently, according to the characteristics of the network and the nature of the underlying links.

We argue that the strategy module answers these two questions in the context of its specific network environment, and therefore bridges the information and connectivity planes. Allowing a spectrum of strategies to co-exist under the umbrella of the ICN architecture provides flexible forwarding behavior that can be adapted to the characteristics of the local connectivity. Hence, an application asks "What is the content for this name?" in the information plane, and a strategy relies on a set of input considerations in the connectivity plane when answering the questions of "Who might have the content?" and "How to retrieve it?".

We specify the forwarding strategy as the architectural component that bridges the information and connectivity planes in ICN. Moreover, we argue that choosing the right mechanism when moving between the information and connectivity planes — the role of the forwarding strategy — is a key element in the design of ICN, and what makes ICN operate in both Internet-like infrastructures and dynamic, non-stable topologies where current Internet methods do not work [11–15].

Although the design and choice of specific mechanisms to bridge the two planes in any given circumstance is a very interesting problem, this work focuses on resolving the tensions created because forwarding strategies, as presently defined, manage the interests of both application and network operators.

According to our definition of its architectural role, it is clear that every forwarding strategy mechanism must consider network characteristics. Therefore, in order to decouple information and connectivity, we argue that forwarding strategies should not implement information-oriented mechanisms, but should contain only connectivity-related mechanisms. When decoupled from application-level mechanisms, forwarding strategies can be safely chosen and deployed by network operators, according to the connectivity they manage.

To summarize this section, we illustrate the questions ICN must answer when translating its abstraction into a set of practical network protocols. We specify the forwarding strategy as the component that today answers those questions with respect to local connectivity characteristics, and therefore bridges the information and connectivity planes in ICN. As a result, we argue that forwarding strategies should implement connectivity-oriented mechanisms, and be decoupled from information-oriented mechanisms. Therefore, forwarding strategies should be selected by network operators according to the connectivity they manage, and should not be paired to namespaces by application developers.

4 INFORMATION-CENTRIC TRANSPORT

Specifying the role of forwarding strategies leads to the following conclusion: forwarding strategies should not be exposed to applications, and should not implement application-level mechanisms. Decoupling information-oriented mechanisms from the strategy component is one step toward decoupling information and connectivity. But it is not enough.

By eliminating the in-network application-level mechanisms previously implemented by forwarding strategies, we transfer those mechanisms to the end-points. It is true that, just as in IP, those mechanisms could be handled by end-point libraries and be decoupled from applications. However, just like HTTP-based applications, ICN applications would still need to respond to connectivity events, such as a change of interface or packet loss.

Consider the following example: Can an ICN application retrieve new data when there is never a synchronous end-to-end path between a consumer and a producer? As we show in Section 7, the current answer is no. In-network caching allows applications to retrieve previously consumed data, but is not useful if the data is new or has already expired. Therefore, while ICN is channel-less, applications are still bound to the end-to-end communication model. We believe that name-based strategy selection was created to address this challenge. However, as discussed in Section 3, coupling information and connectivity mechanisms in the same architectural component is unscalable, and we must find another solution.

As a remedy, we propose *Information-Centric Transport (ICT)*. We define ICT as both an abstraction and a communication mechanism that allows applications to operate solely in the information plane, dealing with namespaces and the trust relationship, while remaining free from connectivity concerns. An ICT is designed to support a specific, but broadly applicable, set of application requirements. Figure 2 shows that an ICT consists of two components: an API for applications at the end hosts, and an intermediate service that runs on selected devices in the network.

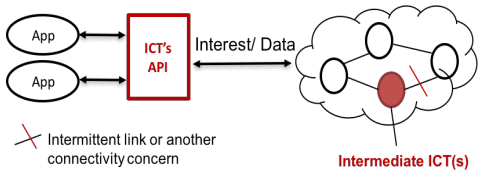

Figure 2: ICT as a two-component transport in ICN: A library API at the end-point, and an intermediate process in the network

To illustrate the concept of ICT, consider how it relates to traditional notions of transport. Existing transport concepts can readily be seen in the IP protocols, which can be viewed as *Connection-Centric Transport*.

- Connection-Centric Transport (CCT): concerned with end-points and channel characteristics, such as reliability and in-order delivery.
- Information-Centric Transport (ICT): concerned with data names and the trust relationships between named identities.

The properties of a CCT are channel-based, and CCTs such as TCP and UDP enable applications to meet different reliability requirements. An IP-based application can also implement its own transport mechanisms by following the Application-Level-Framing (ALF) concept [16]. Related work has shown that ICN can provide similar transport mechanisms to applications with CCT requirements [17–19]. ICT does not preclude CCT transport mechanisms for ICN applications. Instead, ICT extends the concept of transport

to include new in-network and name-based transport mechanisms for applications that want to stay in the information plane, and are not concerned only with CCT properties.

The intermediate ICT process is deployed by the network operator where connectivity characteristics require it (such as intermittent links or in dynamic and mobile networks). When deployed in the network, an ICT must address information plane requirements with respect to existing connectivity characteristics and network mechanisms, while the forwarding strategy addresses concerns raised by the connectivity plane. Therefore, ICT is information-oriented, while the forwarding strategy is connectivity-oriented. For instance, an ICT can buffer interest packets and store data packets if needed for resilient data delivery, and forwarding strategy can add, remove, or probe faces in response to connectivity changes.

But how can ICN provide scalable in-network transport? The end-to-end principle [20] determines that, for scalability, application-specific features should remain at the end nodes and never reside in the network. Although ICN already maintains an in-network state in its PIT and CS, we argue that, to ensure scalability, an ICT should never implement any application-specific mechanisms. Therefore, an ICT must be implemented to capture abstractly a specific set of application-level needs, and for scalability reasons, those needs must be shared among different types of applications.

We believe that future ICT mechanisms can address a substantial range of abstract application needs, from purely semantic to performance and reliability. However, at this point, to the best of our knowledge, there is no clear understanding of the true needs and requirements of ICN applications. Exploring the different sets of application-level requirements that should be implemented by different ICTs is a rich area for future work, and we anticipate that a number of widely useful ICTs may emerge over time.

In this paper, we demonstrate the ICT concept by providing two examples of ICTs that were designed to support simple application semantics and that can be expressed by namespace operations. ICT-Sync, described in Section 5, is an enhancement of NDN sync, and was designed to retrieve all the names under a specific application prefix. ICT-Notify, described in Section 6, was designed to retrieve only the latest name pushed under a namespace. Although seemingly simple, ICTs for these two namespace operations can support a wide range of applications and therefore serve as ICN primitives.

Once a primitive ICT has been created, an application can choose it by using its end-point API. This API is then responsible for expressing Interest and Data packets to satisfy the application-level needs. Furthermore, as a distributed service, every ICT must clearly define its naming convention so that an intermediate ICN router can recognize the application's ICT requirements.

Figure 3 illustrates a network where an ICT can be provided to support applications. In the figure, Alice, Bob, and Ted are participants in a topology that does not always have a synchronous end-to-end path. In this specific example, if link 1 and link 2 are never up at the same time, there will never be an end-to-end path between Alice and the other participants. Therefore, existing end-point NDN libraries cannot forward to Alice any Data packet from other participants. However, if Alice uses an ICT API, an intermediate ICT process deployed in the network can provide Alice with continuous transport despite of the links state. ICT evaluation of similar topologies is shown in Sections 7.

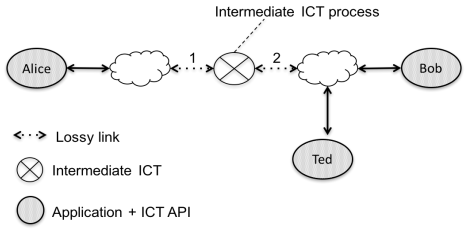

Figure 3: ICT in a Lossy Environment

To conclude, we define an ICT as a new abstraction proposed for applications, one that decouples the information and connectivity planes without knowing or relying on any application-specifics. An application uses the ICT API, and the intermediate component of an ICT is placed in the network by the operator where connectivity characteristics and network mechanisms might interrupt the request for the named data abstraction. To illustrate the ICT abstraction, we discuss two proof-of-concept ICTs in Sections 5 and 6. We show how by operating in the Information plane at the end hosts, and by operating in the intersection of the information and connectivity planes in the network, ICT-Sync and ICT-Notify can make an application agnostic to connectivity details, even when there is never a synchronous end-to-end path.

5 ICT FOR SYNC APPLICATIONS

Sync has been discussed in the past as an ICN primitive [21], and therefore is a great candidate for enhancement as an ICT. In this section we discuss *ICT-Sync* as a secured ICT that supports sync-based applications in lossy environments. We do not propose a new scalable synchronization protocol, but simply demonstrate the concept of ICT and show how ICT-Sync supports sync-based applications to function in environments that might have unstable links. The placement of the intermediate component of ICT-Sync in the network is entirely up to the network operator, and applications written to use ICT-Sync API can function without it if links are stable. The intermediate ICT-Sync component allows such applications to function in lossy networks without any modifications to the application.

5.1 Sync in ICN

The Custodian-Based Information Sharing (CBIS) system [21] was the first implementation of an ICN-based sync service. In this early paper, the authors discussed the high-level principles of what later became the foundation of other ICN sync protocols.

In brief, sync can be described as a process that provides data consistency of shared namespaces over time. Although in this paper we discuss sync in the context of ICN applications, sync's premise is widely used today by many IP-based applications, such as BitTorrent Sync, DropBox, Google Drive, and more. However, while an IP application has to design and implement its own sync paradigm to support the type of its shared data, ICN sync synchronizes namespaces that can represent any type of data. Therefore, sync is a primitive in ICN, and can be used by different type of applications, such as distributed file sharing[22, 23], chat applications[24], and routing protocols[25].

The following sync services for ICN have been proposed in related work: ChronoSync [26], CCNxSync [27], iSync [28], and PartialSync[29]. While these protocols can be differentiated by their implementation details, including their namespace design, mechanism, and data structures, they all follow the same high-level goal of providing a continuous synchronization of namespaces. In this section we discuss only the relevant background details of ChronoSync[26], as it is the sync protocol we used for our ICT-Sync.

ChronoSync is an end-point sync service library implemented in NDN. A participant in the synchronization protocol uses ChronoSync API to register both sync and data prefixes. The sync prefix is shared among all the participants of an application. The data prefix is unique to a party, and it is the prefix of all future data names to be published by that party. The full name of every published data consists of the data prefix and a sequence number. ChronoSync maintains the state of the synchronized parties using a sync tree structure that stores the latest known published sequence number per participant. A root digest is calculated on the names and sequences to represent the state of a particular sync tree. Two sync trees of the same sync prefix are considered up-to-date if their root digests are equal.

Periodically, to notify remote parties about the status of the sync tree, ChronoSync triggers a long-lived sync Interest consisting of the sync prefix and the root digest of the local sync tree. Upon receiving a sync Interest, a party compares the received digest with its local root digest to determine whether its current knowledge about of the shared set is up-to-date. If a received root digest is older than the local one, i.e., it can be found in the digest log table, then ChronoSync responds with a Data packet containing the missing sequence numbers and their corresponding data prefixes. Upon receiving a sync Data packet, ChronoSync updates the local sync tree with the missing sequence numbers and notifies the application. The application can then fetch the content for the new number by expressing an interest for the missing sequence number(s).

5.2 ICT-sync

As an enhancement of ChronoSync, ICT-Sync uses the same namespace design and sync tree structure to represent the prefix of the synchronized dataset. However, unlike ChronoSync, ICT-Sync has an intermediate component of that operates as a separate process. The intermediate ICT-Sync process can be added to any NDN router and acts as a producer of the data published under the sync name. As such, it can be deployed in places in the network where the operator anticipates lossy or unreliable communication. This way, ICT-Sync provides continuous sync service for applications, regardless of the quality of the underlying links. ICT-Sync API is simple, and includes three major calls: 1) define prefixes for synchronization; 2) publish a new content; and 3) define application's trust model.

After the intermediate ICT-Sync process is deployed in the network, it receives and responds to updates from participants in the sync tree. We implemented the intermediate process to automatically fetch the content associated with updates. On receipt of content, it validates the signer of the content, using its trust model. If the content is validated, it saves the full Data packet, including the original signature of the content and all headers. Then, it serves as a provider of the fetched data by registering the participant's prefix in its local NFD. Our implementation can be configured to use either a persistent storage or the NDN CS to store the fetched data, depending on the characteristics of the network and the router.

The characteristics of ICT-Sync as Information-Centric Transport are as follows:

- ICT-Sync is a primitive and it can be used by any application that is looking for sync-based services.
- The existence of the intermediate ICT-Sync component does not introduce any change to the application running at the endpoints.
- The intermediate ICT-Sync component has no knowledge of any application-specifics property or the content it handles. It builds its local sync tree from looking at the names of Interest and Data packets and without decrypting the content.
- Because an application uses the same API in both reliable and challenged networks, its information plane is fully decoupled from the actual connectivity plane.
- ICT-Sync maintains existing NDN trust schema mechanisms to fetch keys and validate the data. It does this by using the existing NDN tools and by looking at the packet's name and key-locator fields, without decrypting the content.

6 ICT FOR NOTIFICATION-BASED APPLICATIONS

In this section, we explore a new NDN service, a notification service, which we believe can become another primitive for ICN applications. We define ICT-Notify to be a push-based mechanism that pushes the latest data under a specified prefix. Just like ICT-Sync, ICT-Notify can support different types of applications. However, unlike ICT-Sync, the goal of ICT-Notify is not to maintain data consistency over time, but to guarantee that the latest content of every data producer is pushed to every consumer. Examples include sensors in an IoT network that want to push their latest measurement to others, AR/VR applications that request to push events' occurrences to other participants in a game, and traditional applications like chat that simply want to notify others of statuses such as "online"/"offline"/"away".

One could say that ICT-Sync can be used to satisfy the same application requirement because it synchronizes the entire set of names, and therefore it is guaranteed to synchronize the latest. However, we argue that the mechanisms required to support full namespace synchronization are heavyweight for applications that need only the latest update(s). For instance, sync-based applications must spend an additional Round-Trip-Time (RTT) [26–29] to fetch the content because sync retrieves only the announcement of the new content and not the data itself.

To illustrate the differences, consider a simple application that wants to present a connectivity sign for each participant: A green circle when a participant is 'on', a yellow circle when he is 'away' and a red circle when he is 'offline'. Using ICT-Sync, the application gets all the connectivity statuses. Using ICT-Notify, the application gets the latest status. For this specific application, in the case of a disconnected or lossy environment, it might make more sense to

use ICT-Notify and get the latest status quicker rather than getting all the statuses pushed when a link was down.

In addition, while pub-sub mechanisms [29] might seem appropriate as notification services, their focus is on retrieving large streams of data from specific known sources, rather then small notifications from multiple dynamic sources. Therefore, they require configuration and do not support dynamic environments. In this section, we explore an approach for a dynamic multi-source notification service, with the goal of pushing small notifications within a one-way delay latency.

6.1 Notifications in ICN

Recent ICN works have studied the problem of handling push-based notifications. [30] discusses and evaluates three schemes to support push-based traffic in NDN: Interest notification, unsolicited data, and virtual Interest polling. Interestingly, due to the trade-off between a device's efficiency and the network overhead, it was found that no scheme is better than the others, and that the selected scheme should be decided according to the expected traffic load and the constraints of devices. [31] discusses a framework for multi-source data retrieval in IoT networks. This framework uses exclude filters to allow selective data retransmissions, and controls PIT deletions to support multiple Data packets for the same Interest. [32] proposes a new ICN packet type for push notifications.

The work in [31] and [32] propose changes to the ICN architecture to better support push notifications. In contrast, our work focuses on implementing a proof-of-concept multi-source notification ICT without any modifications to the architecture. Our goal is to show that such an ICT can be a primitive in the current implementation of NDN, without any modification to the PIT or the forwarder, and without adding a new packet type. We do not argue that our work is more efficient then the alternatives. Rather, we argue that a mechanism for push-based notifications can evolve as an ICT because it supports a common application need, and it can allow NDN applications to stay in the information plane, free from connectivity concerns.

6.2 ICT-Notify

For brevity, we discuss only the high-level details of ICT-Notify. ICT-Notify should not be considered as a final state notification service mechanism, but as a proof-of-concept ICT mechanism for pushing the latest data of a producer. As mentioned, ICT-Notify is not designed to support data consistency over time, but instead, to push only the latest X notifications, where X can be defined by the application, and within one-way delay latency. Therefore, applications that use ICT-Notify must tolerate the loss of past notifications if new are available. We define the following requirements of ICT-Notify to ensure that it is a primitive ICT that can be used by different types of ICN applications:

- Supports multi-source data retrieval.
- Introduces a simple API for applications.
- Support scenarios in which the exact number and identity of producers is unknown.
- Does not require changes to the ICN architecture or forwarders.
- Maintains trust relationships as defined by applications, and does not require to decrypt an application's content.
- Supports different types of applications, without relying on any application specific.

ICT-Notify consists of an application library and an intermediate process that can be deployed by the network operator. The implementation of ICT-notify follows the long-lived Interest scheme. As discussed in [30, 32], this scheme presents challenges in namespace design. In short, although multicasting an Interest with a general name to all potential producers consumes one PIT entry, it does not support simultaneous multi-source Data packets because the first Data packet consumes the PIT entry. In contrast, although sending a designated Interest to each producer supports simultaneous Data packets, it consumes more network overhead (state and bandwidth). To address our requirement for dynamic environments with unknown identities of producers, we choose the first approach and use a general namespace design. ICT-Notify supports simultaneous data delivery through its namespace design, with the penalty of an additional latency. As our focus is to implement a proof-of-concept ICT mechanism, this work does not study the cost of long-lived Interests on the PIT, and does not evaluate the tradeoffs.

Another challenge of the long-lived Interest scheme is to distinguish one Interest from another. In the past, this has been done by adding a sequence number or a timestamp to the Interest name. A sequence number is used to request the next sequenced notification, and a timestamp is used to request a notification that occurred after that time. However, none of these options provide a complete solution. Sequenced Interests require the producer's identity in the namespace, and timestamps do not support unordered data delivery. For instance, consider that consumer C sends a long-lived Interest looking for notifications that occur after t1. Producer P1 responds with a notification that occurred at t2, and a little after, producer P2 responds with another notification occurred at t3. If t3 arrives at C before t2, C will then ask for notifications that occurred after t3, and will miss t2. ICT-Notify takes a new namespace design approach to address this challenge.

To address these challenges, ICT-Notify follows the next namespace design for its long-lived Interest notifications: *<ICTName>/<AppName>/<ConsumerState>*. <ICTName> identifies ICT-Notify packets in the network. <AppName> differentiates applications that use ICT-Notify and supports multicasting to only application parties. <ConsumerState> is the list of recent notifications known by the consumer. A ***notification list*** is a tuple of pairs. Each pair consists of a unique node id, and the unique ids of the last X events pushed by that node.

Determining the best representations of these unique identifiers require further exploration. As a proof-of-concept mechanism we use a random number, generated once by ICT-API, as a node UID, and a timestamp as the event UID. When using the timestamps representation, the ICT-Notify library on the producer side can not determine only what events are missing, but also if the missing events have not expired. In future work, we plan to explore additional identifier representations.

We define an ***event*** to be the content of a pushed Data packet, in the form of a short name, for instance: "/sensorA/temperature/50" or "/User/Alice/Location/X/Y". Since the event is the Data payload,

and not the Interest or Data name, it can be of any form, and it can contain producers' identities if desired by the application. When an applications pushes an event, ICT-Notify API generates a unique event identifier, i.e., a timestamp in our proof-of-concept implementation, and associates it with the event. To minimize collisions of frequently generated events, ICT-Notify uses nanosecond accuracy for timestamps.

The name of a notification Data packet consists of the Interest name and an additional name component, <ProducerState>, that lists the latest notifications known by the producer. Before a new long-lived Interest is sent, old notifications are removed from the <ConsumerState> name component.

When receiving an Interest, a producer compares its local list of known event identifiers with the identifiers in the received <ConsumerState> name component, and determines if it has new events unknown to the consumer. If so, the producer responds with a Data packet that consists of 1) its local notification list in the <ProducerState> name component, and 2) the events that correspond to the missing identifiers in the payload. Therefore, the Data name contains two notification lists: the consumer's out-of-date list from the Interest name, and the latest up-to-date producer's list. The Data payload contains the set-difference of the two lists, which is a list of the missing events. To be clear, the list in the name consists of event identifiers, and the payload consists of the encrypted events.

Since the long-lived interest scheme guarantees that there is always an Interest in the PIT, a producer can push Data notifications whenever the application generates an event. Having <ConsumerState> in the Interest name allows every participant, including the intermediate ICT process, to find the relevant set-difference and to respond quickly with Data. It also enables relevant data retrieval from NDN caches.

We explored two implementations of <ConsumerState> and <ProducerState>: an invertible Bloom filer (IBF) [33], and a simple vector. We encoded each of these data structures in a name component and compressed these names using the Bzip algorithm. We found that although IBFs are considered to be efficient data structures, they consume more memory than vectors when supporting small numbers of items (in the hundreds). Our experiments showed that Interest names with vector representations are six times smaller than Interest names with IBF representations. A quantitative evaluation of vectors is presented in Section 7.2. Since the goal of ICT-Notify is to push only the latest notifications, old names are removed from the vector and therefore we do not anticipate a large number of events encoded in the name components.

Furthermore, ICT-Notify API allows applications to define filters on the events they request to receive. This way, different instances of the applications can choose to be notified about specific events, and not about every event generated by a producer. For instance, a sensor in an IoT network can wake up and send its temperature periodically, but an application can choose to be notified only when the temperature is above X or below Y. ICT-Notify follows the implementation of the schematized trust model [34] to enable regular expressions as configurable filters, as is done to define trust relationships.

The intermediate process of the ICT is similar to an end-point consumer. It does not produce notifications, and unlike ICT-Sync, it does not need to store all Data packets. Instead, the intermediate ICT process maintains a local map of an application's notification list, the tuple of producers and events, according to the flow of Interest and Data names it sees. It keeps track of the latest events pushed by each consumer, and saves only the corresponding payloads. Then it responds to Interests just like any end-point party, without the need to decrypt the data. This way, ICT-Notify supports applications even when the network is disturbed, even in the worst case of no synchronous end-to-end path.

To summarize, ICT-Notify supports applications in two aspects: 1) It proposes a push-based mechanism as a primitive for ICN applications, without relying on any application specifics. 2) Like ICT-Sync, it decouples information from connectivity and supports applications in different connectivities, even if there is no synchronous end-to-end path.

7 DEMONSTRATING ICT-SYNC AND ICT-NOTIFY

We implemented ICT-Sync and ICT-Notify and demonstrated them in a lossy environment, where an end-to-end communication was not always guaranteed. Figure 4 illustrates the NDN topology we used, which consisted of six end-points and four intermediary NDN routers.

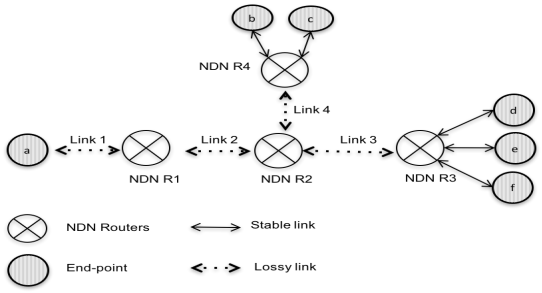

Figure 4: Tested NDN Topology

Interestingly, although it seems simple, this topology illustrates an use case of a lossy environment that disturbs application flow. For instance, consider a sensor network in which two sensors communicate via one or more intermediate nodes and links that can asynchronously move or fail. As we show in this section, without an ICT, the application completely breaks using IP-based or NDN end-to-end services, but can successfully operate when supported by an ICT.

We conducted all our experiments on the Open Network Lab (ONL) [35], where real routers and links can be programmed to control the factors we used in our experiments: link delay, link bandwidth, packet drop rates, and link availability. We used 10 two-core machines as end-points and NDN routers, and five Ubuntu Linux (16.04.4) software routers servers. All two-core machines ran NFD [7]. Each experiment was repeated at least five times in order to control experimental error caused by the dynamics of the network. The end-points ran tested applications with ICT-sync and ICT-Notify APIs. NDN routers were a combination of two machines - a Linux software router and a machine that ran the NFD code.

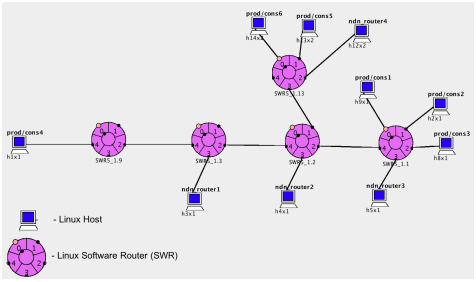

Figure 5: Mapping Topology onto Physical ONL Hardware

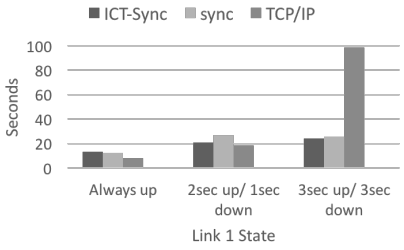

Figure 6: Fetch Time with Different Communications with Different Link state

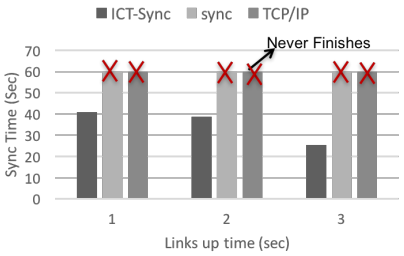

Figure 7: Fetch Time with Different Communications with Alternating Links

7.1 Demonstrating ICT-Sync

The goal in this section is to show the capabilities of ICT-Sync, rather than the scalability of chronoSync. Therefore, in these experiments, we focus on demonstrating how ICT-Sync provides transport for a file sharing application by decoupling it from connectivity characteristics.

In all ICT-Sync experiments, node 'a' encrypted data read from a local 1MB file into 1KB chunks, and published it using ICT-Sync. Following ChronoSync protocol, the ICT-Sync API updated the sync tree to represent the sequence number of the new chunk. The sync libraries on end-points 'b'-'f' exchanged sync Interest and Data packets to reconcile the new data, and notified the application of the new chunk name. In our experiments, we implemented the application to fetch every new chunk in order to measure its performance on top of a variety of connectivities. Once the chunk was received, the signer was validated, and the content was decrypted, and concatenated into a single file. ICT-Sync also followed the application trust model by validating every received data packet. In our experiment, we preconfigured the nodes with the sync prefix for the content and the trust anchor needed for validation.

We used the consumers' system clock to record the start time, when it received the first ChronoSync update, and the end time, when it finished fetching the last chunk of the synced file. We set the Link rate to 1000Mbps and manipulated the experimental factors to replicate an ad-hoc network with low bandwidth and intermittent links. The calculated error in all the experiments was found to be very small relative to the measured value, and therefore did not affect the reported results.

In the next experiment we measured the average fetching time of a file sharing application with and without ICT-Sync deployed in the network. Following the ICT concept, we ran the same application on the different setups. When using ICT-Sync, we configured it to run on R1. In addition, we tested the same topology with a similar IP-based application by using ipref. We recorded how many seconds elapsed from the moment the consumer discovered a new chunk until it fetched the entire file.

Figure 6 shows the consumer fetch time on top of different communications in three different states: 1) Link 1 is always up, hence there is a synchronous end-to-end path between the consumers and the producer . 2) Link 1 is up for two seconds and down for one second. and 3) Link 1 is up for three seconds and down for three seconds. ICT-Sync denotes the setup in which the intermediate ICT runs on router 1. TCP/IP denotes similar IP-based application by using ipref, and sync indicates the setup in which only the end hosts run Sync.

Figure 7 shows the consumer fetch time in the scenario in which link 1 and link 2 alternated for different amounts of time. Here, there is never an end-to-end path, because we stopped one link before we woke up the other. In this test the x-axis indicates the number of seconds each link is up before being stopped. The results in Figure 6 and Figure 7 support the following conclusions:

- The application succeeded in fetching the 1MB file when links 1 and 2 broke the end-to-end path only when ICT-Sync was running on router 1. Without ICT-Sync, the consumer failed to fetch the file
- ICT-Sync does not improve performance when the end-to-end path is reliable and un-interrupted.

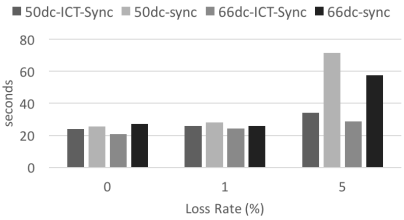

Figure 8: Fetch Time with Different Packet Loss Rates

In the next set of experiments, we configured our application to synchronize a 1MB file on top of different connectivity characteristics. We measured the average fetch time with different connectivity

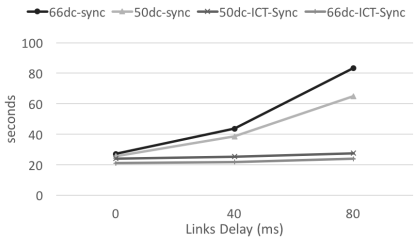

Figure 9: Fetch Time with Different Links Delay

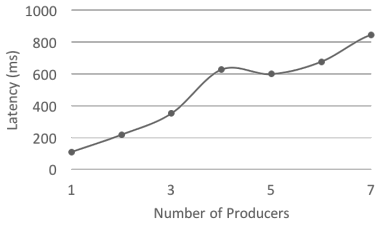

Figure 10: Push Time (ms)

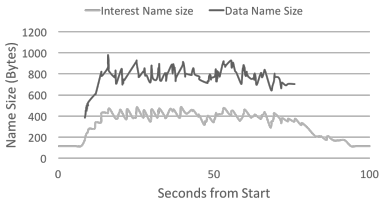

Figure 11: Name Size (Bytes)

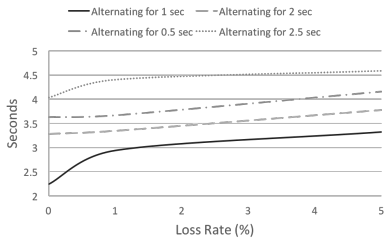

Figure 12: Push Time with no End-to-End Path (sec)

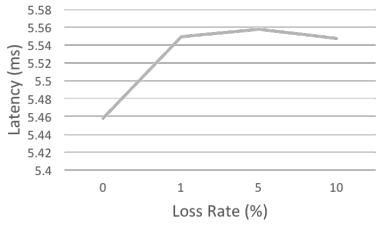

Figure 13: Median Push Latency over Different Loss Rates (ms)

characteristics, such as link delay and packet drop rate, and modified the duty cycle of link 1. A duty cycle of 50% indicates that the link was up and down for three seconds each time. A duty cycle of 66% indicates that the link was up for two seconds and down for one second. The results are presented in Figures 8 and 9.

The results support the following conclusions:

- With no consistent end-to-end path, running ICT-Sync on node 1 always expedited fetch times in the tested scenarios of different link delays and different packet losses.
- For a 50% duty cycle, the application can fetch up to 57% faster with ICT-Sync.
- For a 66% duty cycle, the application can fetch up to 72% faster with ICT-Sync.

7.2 Demonstrating ICT-Notify

The experiments discussed in this section focus on showing the capabilities of ICT-Notify as a primitive, rather than its scalability. Scalability tests as well as more technical details will be published separately. We used the NDN topology in Figure 4 over the physical topology in Figure 5. Network Time Protocol (NTP) was used on ONL machines to ensure the consistency of timestamps.

Figure 10 shows notifications' latency for different number of producers. In this set of experiments, we increased the number of producers that send simultaneous notifications. Each producer sent between 80-100 notifications every 1-4 seconds. We measured the time elapsed from the moment a producer pushed a notification until it arrived at the consumer. We used node 'a' as the consumer, and nodes 'b'-'f' as the producers. The results suggests a linear trend between the number of simultaneous updates and the push latency. This linear trend represents the penalty of using the long-lived Interest scheme, because only one Data packet consumes the Interest in the case of simultaneous updates. The following Data packets are aggregated at the intermediate ICT and wait for the next long-lived Interest.

In the next sets of experiments we deployed the intermediate ICT-Notify process on NDN R1. Figure 11 shows the name size (in Bytes) of the Interest and Data packets sent and received by the intermediate process. We programed five producers to start sending simultaneous notifications approximately four seconds after we started each experiment. We set notifications to expire five second after they have been sent. Each producer sent 50 notifications every 1-3 seconds. As shown by the figure, Data names are approximately twice the size of Interest names as they carry both the consumer and producer timestamps lists. Figure 10 also shows how the name size drops when expired timestamps are removed.

Figure 12 shows notifications' latency for different loss rate in the scenario in which link 1 and link 2 alternated for different amounts of times, hence, there is never a synchronous end-to-end path. The x-axis of Figure 12 indicates the different loss rates we tested, and the y-axis indicates the latency for the different up and down times we tried. The results show that notifications are pushed faster when the links alternate for 1 and 2 seconds than alternating for 500 ms. As with ICT-Sync, repeating this experiment

without the deployment of the intermediate ICT process lead to zero notifications received by the consumer. Our experiments show relatively long fetching times (in seconds) that are not competitive for push-based applications. After close examination, we found that NFD randomly blocks our application, and therefore, the results contain a few large outliers. To provide a better evaluation of ICT-Notify, and not NFD performance, we present the median push time of one producer over different loss rates in Figure 13.

7.3 Related Work

Most related works explore different mechanisms for forwarding strategies in ICN. Presently, NFD and CICN both implement forwarding strategies, and each strategy consists of specific forwarding mechanisms. [6, 10]. [36] discusses the principles of an adaptive forwarding strategy, while proposed mechanisms for such a strategy are presented in [37]. A dynamic forwarding mechanism designed to discover temporary copies of content items is presented in [38]. [39] proposes a revised forwarding strategy that can better prevent or detect loops in NDN. Strategies for Wireless networks are discussed in [40, 41], and a set of adaptive forwarding strategies for access networks are described in [42]. [43] presents a probability-based adaptive forwarding strategy, including a statistical model to compute strategy retransmission intervals.

In contrast to these cited works, our work is mainly focused on solving the problem in which the forwarding strategy component couples applications with connectivity. The works in [2, 3] discuss the relation between application and forwarding strategies, but do not attempt to solve the problem. The works in [4, 9] specifically attempt to address the conflicts created by the strategy decision whether to retransmit an unsatisfied Interest.

The work in [17] proposes a consumer-producer API, built to simplify and reduce the implementation efforts of NDN application developers. Although this work addresses some of the challenges discussed here, we propose a general-purpose information-centric abstraction and a mechanism that provide in-network transport. Not only an API. Another implementation of a socket API for ICN is discussed in [18].

The Named Function Network (NFN) architecture [44] supports in-network code execution by applying named functions to named data. The proposed ICT abstraction follows the same approach and allows in-network information-oriented mechanisms, but unlike NFN, it does not require changes to the Interest structure, and therefore to the forwarding plane. However, the tradeoffs between the two approaches should be further explored in future work.

In addition, while related sync works [26–29] explore and evaluate different sync mechanisms for ICN, our proposed ICT-Sync is an enhancement of ChronoSync and is focused on providing transport for sync-based applications. Therefore, our contribution differs from those of other works proposing sync services.

8 CONCLUSIONS AND FUTURE WORK

In this work, we proposed to decouple information and connectivity in Information-Centric Networking by specifying the role of forwarding strategies and by introducing the concept of Information-Centric Transport (ICT). ICT seeks to fulfill the promise of the ICN abstraction — the request for named data — and allow applications the freedom to stay in stay in the information plane focusing on requesting named data and defining trust identities. An ICT implementation consists of an API for application developers, and an intermediate network process for network operators. An intermediate ICT should not contain any application-specific knowledge, and should support ICN trust mechanisms. Moreover, the concept of ICT does not preclude other applications from using traditional end-to-end transport mechanisms.

It may appear that the core contribution of this paper is to allow placing function-specific blobs at intermediate nodes. However, we argue that forwarding strategies already implement function-specific blobs in the network. This paper's core contribution is to decouple information-oriented mechanisms from connectivity-oriented mechanisms by placing the ICT component in the ICN architecture. To make the ICT abstraction practical and scalable, ICN must implement a set of well-defined ICTs. Determining the different sets and the specific application requirements that each set should implement remains a question for future work.

We describe two ICTs to demonstrate the ICT concept: ICT-Sync, an ICT for applications that request full synchronization of their names, and ICT-Notify, an ICT for applications that request to be notified on the latest data of their name. We show that when using these ICTs, sync-based and push-based applications remain entirely in the information plane, agnostic to connectivity characteristics. We further show that applications can thus operate in environments where IP-based applications and present NDN services fail.

In future work, we plan to extend our efforts and explore new ICTs to support other application-level requirements. In addition, our current implementations of ICT-Sync and ICT-Notify were designed only to demonstrate the concept of ICT, and we plan to improve ICT-Sync and ICT-Notify to be more robust and efficient. The demonstrated ICTs use simple namespace discovery, however, it is unclear whether other ICTs, designed to support other application needs, should follow the same method. Exploring how an intermediate ICT discovers its Interest and Data packets is a rich area for future work.

Additionally, the interaction between the ICT component and other network components must be further explored. For instance, how does routing interact with ICTs? Should QoS and congestion control mechanisms be implemented in the connectivity plane (forwarding strategy) or in the information plane (ICT)? Can a single ICT implement different mechanisms for the same set of application-needs? For instance, can we have one universal ICT that supports different sync schemas? Moreover, the exact placement(s) of an intermediate ICT should be explored and better understood. For instance, should it be deployed everywhere, or only in some specific network elements? All these questions and more will be part of future work.

ACKNOWLEDGEMENTS

The authors would like to thank our shepherd John Wroclawski for providing very thorough and constructive comments. The authors would also like to thank the anonymous reviewers for their valuable comments and helpful suggestions. The work is supported by the National Science Foundation (NSF) under the following grants: CNS-1040643, CNS-1345282, CNS-1719366 and CNS-1629807.

REFERENCES

[1] Van Jacobson, Diana K Smetters, James D Thornton, Michael F Plass, Nicholas H Briggs, and Rebecca L Braynard. Networking named content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 1–12. ACM, 2009.

[2] Hila Ben Abraham and Patrick Crowley. Forwarding strategies for applications in named data networking. In *Proceedings of the 2016 Symposium on Architectures for Networking and Communications Systems*, pages 111–112. ACM, 2016.

[3] Daniel Posch, Benjamin Rainer, and Hermann Hellwagner. Towards a context-aware forwarding plane in named data networking supporting qos.

[4] Hila Ben Abraham and Patrick Crowley. In-network retransmissions in named data networking. In *Proceedings of the 2016 conference on 3rd ACM Conference on Information-Centric Networking*, pages 209–210. ACM, 2016.

[5] Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson, Patrick Crowley, Christos Papadopoulos, Lan Wang, Beichuan Zhang, et al. Named data networking. *ACM SIGCOMM Computer Communication Review*, 44(3):66–73, 2014.

[6] Project CICN. https://wiki.fd.io/view/Cicn.

[7] Alexander Afanasyev et al. Nfd developer's guide. Technical report, NDN-0021, NDN, 2014.

[8] NDN Testbed. http://ndnmap.arl.wustl.edu/.

[9] Hila Ben Abraham and Patrick Crowley. In-network retransmissions in named data networking. 2016.

[10] NFD Named Data Networking Forwarding Daemon. http://named-data.net/doc/NFD/current/.

[11] Alexander Afanasyev, Cheng Yi, Lan Wang, Beichuan Zhang, and Lixia Zhang. Scaling ndn routing: Old tale, new design. *Technical Report NDN-0004*, 2013.

[12] Alexander Afanasyev, Cheng Yi, Lan Wang, Beichuan Zhang, and Lixia Zhang. Snamp: Secure namespace mapping to scale ndn forwarding. In *Computer Communications Workshops (INFOCOM WKSHPS), 2015 IEEE Conference on*, pages 281–286. IEEE, 2015.

[13] Alexander Afanasyev, Xiaoke Jiang, Yingdi Yu, Jiewen Tan, Yumin Xia, Allison Mankin, and Lixia Zhang. Ndns: A dns-like name service for ndn. In *Computer Communication and Networks (ICCCN), 2017 26th International Conference on*, pages 1–9. IEEE, 2017.

[14] Xuan Liu, Zhuo Li, Peng Yang, and Yongqiang Dong. Information-centric mobile ad hoc networks and content routing: a survey. *Ad Hoc Networks*, 58:255–268, 2017.

[15] Marica Amadeo, Claudia Campolo, and Antonella Molinaro. Forwarding strategies in named data wireless ad hoc networks: Design and evaluation. *Journal of Network and Computer Applications*, 50:148–158, 2015.

[16] David D Clark and David L Tennenhouse. Architectural considerations for a new generation of protocols. In *ACM SIGCOMM Computer Communication Review*, volume 20, pages 200–208. ACM, 1990.

[17] Ilya Moiseenko, Lijing Wang, and Lixia Zhang. Consumer/producer communication with application level framing in named data networking. In *Proceedings of the 2nd International Conference on Information-Centric Networking*, pages 99–108. ACM, 2015.

[18] Massimo Gallo, Lin Gu, Diego Perino, and Matteo Varvello. Nanet: socket api and protocol stack for process-to-content network communication. In *Proceedings of the 1st international conference on Information-centric networking*, pages 185–186. ACM, 2014.

[19] Peter Gusev and Jeff Burke. Ndn-rtc: Real-time videoconferencing over named data networking. In *Proceedings of the 2nd International Conference on Information-Centric Networking*, pages 117–126. ACM, 2015.

[20] Jerome H Saltzer, David P Reed, and David D Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems (TOCS)*, 2(4):277–288, 1984.

[21] Van Jacobson, Rebecca L Braynard, Tim Diebert, Priya Mahadevan, Marc Mosko, Nicholas H Briggs, Simon Barber, Michael F Plass, Ignacio Solis, Ersin Uzun, et al. Custodian-based information sharing. *IEEE Communications Magazine*, 50(7), 2012.

[22] Alexander Afanasyev, Zhenkai Zhu, Yingdi Yu, Lijing Wang, and Lixia Zhang. The story of chronoshare, or how ndn brought distributed secure file sharing back. In *Mobile Ad Hoc and Sensor Systems (MASS), 2015 IEEE 12th International Conference on*, pages 525–530. IEEE, 2015.

[23] Jared Lindblom, M Huang, Jeff Burke, and Lixia Zhang. Filesync/ndn: Peer-to-peer file sync over named data networking. *NDN, TR*, 12, 2013.

[24] Zhenkai Zhu, Chaoyi Bian, Alexander Afanasyev, Van Jacobson, and Lixia Zhang. Chronos: Serverless multi-user chat over ndn. *NDN, Technical Report NDN-0008*, 2012.

[25] AKM Hoque et al. Nlsr: Named-data link state routing protocol. In *Proceedings of the 3rd ACM SIGCOMM Workshop on Information-centric Networking*, pages 15–20. ACM, 2013.

[26] Zhenkai Zhu and Alexander Afanasyev. Let's chronosync: Decentralized dataset state synchronization in named data networking. In *ICNP*, pages 1–10, 2013.

[27] CCNx Sync. https://www.ccnx.org/releases/latest/doc/technical/SynchronizationProtocol.html.

[28] Wenliang Fu, Hila Ben Abraham, and Patrick Crowley. Synchronizing namespaces with invertible bloom filters. In *Proceedings of the Eleventh ACM/IEEE Symposium on Architectures for networking and communications systems*, pages 123–134. IEEE Computer Society, 2015.

[29] Minsheng Zhang, Vince Lehman, and Lan Wang. Scalable name-based data synchronization for named data networking. In *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE*, pages 1–9. IEEE, 2017.

[30] Marica Amadeo, Claudia Campolo, and Antonella Molinaro. Internet of things via named data networking: The support of push traffic. In *Network of the Future (NOF), 2014 International Conference and Workshop on the*, pages 1–5. IEEE, 2014.

[31] Marica Amadeo, Claudia Campolo, and Antonella Molinaro. Multi-source data retrieval in iot via named data networking. In *Proceedings of the 1st ACM Conference on Information-Centric Networking*, pages 67–76. ACM, 2014.

[32] Ravi Ravindran, Asit Chakraborti, Syed Obaid Amin, and Jiachen Chen. Support for Notifications in CCN. Internet-Draft draft-ravi-icnrg-ccn-notification-01, Internet Engineering Task Force, July 2017. Work in Progress.

[33] David Eppstein, Michael T Goodrich, Frank Uyeda, and George Varghese. What's the difference?: efficient set reconciliation without prior context. In *ACM SIGCOMM Computer Communication Review*, volume 41, pages 218–229. ACM, 2011.

[34] Yingdi Yu, Alexander Afanasyev, David Clark, Van Jacobson, Lixia Zhang, et al. Schematizing trust in named data networking. In *Proceedings of the 2nd International Conference on Information-Centric Networking*, pages 177–186. ACM, 2015.

[35] Charlie Wiseman, Jonathan Turner, Michela Becchi, Patrick Crowley, John DeHart, Mart Haitjema, Shakir James, Fred Kuhns, Jing Lu, Jyoti Parwatikar, et al. A remotely accessible network processor-based router for network experimentation. In *Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, pages 20–29. ACM, 2008.

[36] Cheng Yi, Alexander Afanasyev, Lan Wang, Beichuan Zhang, and Lixia Zhang. Adaptive forwarding in named data networking.

[37] Cheng Yi et al. A case for stateful forwarding plane. *Computer Communications*, 2013.

[38] Raffaele Chiocchetti, Diego Perino, Giovanna Carofiglio, Dario Rossi, and Giuseppe Rossini. Inform: a dynamic interest forwarding mechanism for information centric networking. In *Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking*, pages 9–14. ACM, 2013.

[39] JJ Garcia-Luna-Aceves. A fault-tolerant forwarding strategy for interest-based information centric networks. In *IFIP Networking Conference (IFIP Networking), 2015*, pages 1–9. IEEE, 2015.

[40] Giulio Grassi, Davide Pesavento, Giovanni Pau, Lixia Zhang, and Serge Fdida. Navigo: Interest forwarding by geolocations in vehicular named data networking. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2015 IEEE 16th International Symposium on a*, pages 1–10. IEEE, 2015.

[41] Marica Amadeo, Antonella Molinaro, Claudia Campolo, Manolis Sifalakis, and Christian Tschudin. Transport layer design for named data wireless networking. In *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*, pages 464–469. IEEE, 2014.

[42] Klaus M Schneider and Udo R Krieger. Beyond network selection: Exploiting access network heterogeneity with named data networking. In *Proceedings of the 2nd International Conference on Information-Centric Networking*, pages 137–146. ACM, 2015.

[43] Haiyang Qian, Ravishankar Ravindran, Guo-Qiang Wang, and Deep Medhi. Probability-based adaptive forwarding strategy in named data networking. In *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, pages 1094–1101. IEEE, 2013.

[44] Christian Tschudin and Manolis Sifalakis. Named functions and cached computations. In *Consumer Communications and Networking Conference (CCNC), 2014 IEEE 11th*, pages 851–857. IEEE, 2014.

[45] Marc Mosko. Ccnx 1.0 protocol specification roadmap. 2013.

[46] Project CCNx. http://www.ccnx.org/.

[47] Cheng Yi et al. On the role of routing in named data networking. In *Proceedings of the 1st international conference on Information-centric networking*, pages 27–36. ACM, 2014.

[48] Alberto Compagno et al. To nack or not to nack? negative acknowledgments in information-centric networking. *arXiv preprint arXiv:1503.02123*, 2015.

[49] ndn traffic. https://github.com/named-data/ndn-traffic-generator.

[50] Ze'ev Lailari, Hila Ben Abraham, Ben Aronberg, Jackie Hudepohl, Haowei Yuan, John DeHart, Jyoti Parwatikar, and Patrick Crowley. Experiments with the emulated ndn testbed in onl. In *Proceedings of the 2nd International Conference on Information-Centric Networking*, pages 219–220. ACM, 2015.

[51] Giovanna Carofiglio, Massimo Gallo, Luca Muscariello, Michele Papalini, and Sen Wang. Optimal multipath congestion control and request forwarding in information-centric networks. In *2013 21st IEEE International Conference on Network Protocols (ICNP)*, pages 1–10. IEEE, 2013.

[52] Vince Lehman, Ashlesh Gawande, Beichuan Zhang, Lixia Zhang, Rodrigo Aldecoa, Dmitri Krioukov, and Lan Wang. An experimental investigation of hyperbolic

routing with a smart forwarding plane in ndn. In *Quality of Service (IWQoS), 2016 IEEE/ACM 24th International Symposium on*, pages 1–10. IEEE, 2016.

[53] Minsheng Zhang, Vince Lehman, and Lan Wang. Partialsync: Efficient synchronization of a partial namespace in ndn. Technical report, Technical Report NDN-0039, NDN, 2016.

[54] Lijing Wang, Ilya Moiseenko, and Lixia Zhang. Ndnlive and ndntube: Live and prerecorded video streaming over ndn. *NDN, Univ. California, Los Angeles, CA, USA, Tech. Rep. NDN-0031*, 2015.

[55] Ilya Moiseenko and Dave Oran. Tcp/icn: Carrying tcp over content centric and named data networks. In *Proceedings of the 2016 conference on 3rd ACM Conference on Information-Centric Networking*, pages 112–121. ACM, 2016.

[56] Dave Oran, Mark Stapp, Ilya Moiseenko, and Won So. Representational state transfer operations using information centric networking, February 27 2015. US Patent App. 14/633,580.

[57] Lijing Wang, Ilya Moiseenko, and Dongsheng Wang. When video streaming meets named data networking: A case study. In *High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2016 IEEE 18th International Conference on*, pages 166–173. IEEE, 2016.

[58] Hila Ben Abraham and Patrick Crowley. Controlling strategy retransmissions in named data networking. In *Proceedings of the Symposium on Architectures for Networking and Communications Systems*, pages 70–81. IEEE Press, 2017.

[59] Spyridon Mastorakis, Alexander Afanasyev, Yingdi Yu, and Lixia Zhang. ntorrent: Peer-to-peer file sharing in named data networking. In *Computer Communication and Networks (ICCCN), 2017 26th International Conference on*, pages 1–10. IEEE, 2017.